11-28-00

4

FORM PTO-1082

Case Docket No.P5635
November 27, 2000
Express Mail Label No. EL414495378US
0045.0005

ASSISTANT COMMISSIONER FOR PATENTS
Washington, D.C. 20231

Dear Sir:

Transmitted herewith for filing is the patent application of

Inventor(s): T. Leong; M. Mallikarjuna; and J. Taylor
For: **METHOD, SYSTEM, PROGRAM, AND COMPUTER READABLE MEDIUM FOR PROVIDING A DATABASE FOR OBJECT ORIENTED OBJECTS**

Enclosed are:

X 7 No. of Sheets of Drawings Sheet(s) of drawings (X informal) + 0 extra copies;
24 pages of Application; 12 pages of specification, 1 page of abstract
X An assignment of the invention to Sun Microsystems, Inc. . (___ Will follow.)
___ An associate power of attorney.
___ A verified statement to establish small entity status under 37 CFR 1.9 and 1.27.
X Declaration and Power of Attorney. (___ Will follow.)
___ Certified copy of Patent Application No. filed from which priority is claimed under 35 U.S.C. §119.
___ IDS enclosed. ___ with references.

CALCULATION OF FEES						
	ITEM	NO. OF CLAIMS FILED MINUS BASE*		NO. OF CLAIMS OVER BASE	X SM/LG ENTITY FEE	$ AMOUNT	$ FEE
A	TOTAL CLAIMS FEE	59	- 20* =	39	X $9 or $18	$702	
B	INDEPENDENT CLAIMS FEE**	7	- 3* =	4	X $40 or $80	$320	
C	SUBTOTAL - ADDITIONAL CLAIMS FEE (ADD FINAL COLUMN IN LINES A + B)						1,022
D	MULTIPLE-DEPENDENT CLAIMS FEE SMALL ENTITY FEE = $135; LARGE ENTITY FEE = $270						$0
E	BASIC FEE* SMALL ENTITY FEE = $355; LARGE ENTITY FEE = $710						$710
F	TOTAL FILING FEE (ADD TOTALS FOR LINES C, D, AND E)						$1,732
G	ASSIGNMENT RECORDING FEE					$ 40	$40
	**LIST INDEPENDENT CLAIMS 1, 9, 18, 26, 35, 43, 52						

Please charge Dep. Acct. No. 50-0585 in the amount of $ **A copy of this sheet is enclosed.**

X A check in the amount of $ 1,732 to cover the filing fee is enclosed.
X Check for $ 40 covering the Recordation of Assignment fee enclosed.
X The Commissioner is hereby authorized to charge payment of the following fees associated with this communication or credit any overpayment to Deposit Account No. 50-0585 . **A copy of this sheet is enclosed.**
X Any additional filing fees required under 37 CFR 1.16.
X Any patent application processing fees under 37 CFR 1.17.
___ The Commissioner is hereby authorized to charge payment of the following fees during the pendency of this application or credit any overpayment to Deposit Account No. 50-0585. **A copy of this sheet is enclosed.**
___ Any patent application processing fees under 37 CFR 1.17.
___ The issue fee set in 37 CFR 1.18 at or before mailing of the Notice of Allowance, pursuant to 37 CFR 1.311(b).
___ Any filing fees under 37 CFR 1.16 for presentation of extra claims.

Respectfully submitted,

David W. Victor
Registration No. 39,867

Direct All Correspondence to:
David W. Victor
KONRAD RAYNES & VICTOR LLP
1180 S. Beverly Drive; Suite 501
Los Angeles, CA 90035

Direct Telephone Calls to:

(310) 556-7983

Express Mail No. EL414495378US
Docket No. P5635
Firm No. 0045.0005

METHOD, SYSTEM, PROGRAM, AND COMPUTER READABLE MEDIUM FOR PROVIDING A DATABASE FOR OBJECT ORIENTED OBJECTS

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to a method, system, program, and computer readable medium for providing a database of object oriented objects.

2. Description of the Related Art

An object oriented data base system (OODBMS) provides a persistent and sharable repository and manager of objects defined according to an object-oriented data model. Every object encapsulates a state and behavior. The state of an object comprises the values of the attributes (also referred to as properties) defined for the object, and the behavior of the object comprises the methods provided with the objects. Objects that share the same attributes and methods comprise a class. All the objects maintained in an OODBMS are members of the same class or have the same parent class. This means that the same set of methods defined for the class are used to manipulate the objects in the OODBMS, such as create, delete, add, read, modify, update, etc. Further the objects in a class have the same attributes defined for the class, even though particular attributes within any of the objects in the class may have different values. Objects persistently stored within an OODBMS defined for a class are viewed and distinguished according to the values provided for their attributes. Each object is further provided a unique identifier for use in accessing the object within the OODBMS using the interfaces provided for the class. Benefits and further explanations of object oriented databases are described in"Research Directions in Objected-Oriented Database Systems", by Won Kim (Copyright Association of Computing Machinery, 1990); "Intermedia: A Case Study of the Differences Between Relational and Object-Oriented Database Systems", by Karen E. Smith, Stanley B. Zdonik, OOPSLA '87 Proceedings (Copyright Association of

Computing Machinery, 1987); and U.S. Patent No. 6,128,771, all of which publications and patents are incorporated herein by reference in their entirety.

Currently, many object oriented database systems are implemented using a Java application programming interface (API).** The application programmer may write APIs in Java to use to access the object oriented database management system (OODBMS). The interfaces to the objects in the OODBMS are also written in Java, and the objects in the OODBMS are implemented as Java classes. In such Java implemented OODBMS, Java applications can generate Java objects to add to the Java OODBMS and utilize Java APIs to manipulate the Java objects in the Java OODBMS.

One challenge with prior art object oriented database systems is that applications written in different programming languages cannot share objects in the same OODBMS. For instance, a C or C++ application program creating a C or C++ data object cannot add objects to a Java OODBMs because of differences in the naming conventions and structures in the different programming languages. Thus, although two applications written in different languages may utilize the same class of objects having the same attributes and attribute values, and desire to share the same data objects, the applications in the different programming languages cannot store and access objects in the same OODBMS. Due to such limitations, duplicate object oriented databases must be provided for the application programs in the different programming languages even though such application programs intend to use the same data objects instantiated from the same class. Further, one application will not be able to access or manipulate the objects created by the other application and maintained in that applications OODBMS.

For these reasons, there is a need in the art to provide mechanisms to allow application programs in different programming languages to utilize the same OODBMS.

SUMMARY OF THE DESCRIBED IMPLEMENTATIONS

Provided is a method, system, program, and data structures for maintaining a database of objects. At least one structured document is received representing an instance of an object including attributes and attribute values defined for a class. Content of the structured document representing the object is added into the database, wherein the database is capable of storing multiple structured documents representing multiple objects.

In another implementation, an instance of at least one object including attributes and attribute values defined for the class is generated. For each generated object, a structured document is generated representing the object and including a representation of the attributes and attribute values in the object. Each structured document is transferred to the database to maintain.

BRIEF DESCRIPTION OF THE DRAWINGS

Referring now to the drawings in which like reference numbers represent corresponding parts throughout:

FIG. 1 illustrates a computing environment defining one possible implementation of the invention;

FIG. 2 illustrates an example of a database of objects implemented in an XML file in accordance with certain described implementations of the invention;

FIG. 3 illustrates logic to transform an object to an XML document in accordance with certain described implementations of the invention;

FIG. 4 illustrates logic to transform an XML document representing an object to an object in accordance with certain described implementations of the invention;

FIG. 5 illustrates logic to add an XML document representing an object to a database in accordance with certain described implementations of the invention;

FIG. 6 illustrates logic to generate a list of object identifiers (OID) of objects in the database in accordance with certain described implementations of the invention; and

FIG. 7 illustrates logic to access an object in the database in accordance with certain described implementations of the invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

In the following description, reference is made to the accompanying drawings which form a part hereof and which illustrate several embodiments of the present invention. It is understood that other embodiments may be utilized and structural and operational changes may be made without departing from the scope of the present invention.

FIG. 1 illustrates a computing environment in which certain described implementations operate. Two client systems 2 and 4 communicate with a database server 6 over a network 8 using a communication protocol, such as the Transfer Control Protocol/Internet Protocol (TCP/IP), or other communication protocol known in the art. The database server 6 manages access to an object oriented Extended Markup Language (XML) database (XMLOOD) 10. The XMLOOD 10 is maintained in a storage device accessible through the database server 6. All objects in the object oriented XML database (XMLOOD) 10 are members of a same parent class. The client systems 2 and 4 and database server 6 may comprise any microprocessor computing system capable of providing a platform to execute the programs and instructions described herein, such as a personal computer, laptop computer, server, mainframe, workstation, hand held computer, telephony device, etc. The network 8 may comprise any network connection known in the art, such as a local area network (LAN), Intranet, the Internet, Wide Area Network (WAN), etc., or a wireless connection.

Client system 2 includes an application program 12, which for purposes of description, is implemented in a structured or object oriented programming language other than Java, e.g., C, C++, Smalltalk, Fortran, etc. The application program 12 is capable of producing application data objects 14 that conform to a particular class structure implemented in the application program 8, and are non-Java objects. An

extensible markup language (XML) translator 16 is capable of transforming the content of the data object 14 into a structured XML file 18.

The client system 4 includes an executing Java application 20 executing in the client system 4 that is capable of producing Java objects 22 in manner known in the art. An XML translator 24 transforms the content of the Java object 22 into a structured XML document 26. The client system 4 would further include a Java Virtual Machine (JVM) to convert Java bytecodes to instructions in the native machine language of the client 4. The client systems 2 and 4 are capable of transmitting a data stream to the database server 6 over the network 8 including the XML document 18, 28, and other protocol information.

The database server 6 includes a database daemon 24 that monitors a port on the database server 6 for requests from clients 2 and 4 to access the object oriented database 10. Object oriented database application interfaces (OOD APIs) 30 comprise instructions to manipulate the data in the XMLOOD 10. The OOD APIs 30 would comprise instructions to parse and perform operations within the XMLOOD 10, such as instructions to add data, delete data, access and read data, update data, etc. Such OOD APIs 30 may include typical commands to parse and process an XML file, such as the type of commands found in the Simple API for XML (SAX) parser. Alternatively, the XML files can be generated into a Document Object Model (DOM) tree in a manner known in the art and then manipulated using DOM commands. The OOD APIs 30 are specifically designed to handle and process XML objects stored in the XMLOOD 10. The database daemon 24 invokes the OOD APIs 30 to perform the client requested manipulation of the XMLOOD 10. In implementations where the OOD APIs 30 comprise Java commands or the database daemon 24 is implemented as a Java program, then the database server 6 would include a Java Virtual Machine (JVM) to convert Java bytecodes to instructions in the native machine language of the database server 6.

The XML translators 16 and 26 are capable of parsing a data object in a structured or object oriented programming language and converting the attributes of the object to

tagged elements in an XML file. Each attribute and attribute value in the data object would map to a separate tagged attribute element and tagged attribute value in the XML document representing the object. Such conversions of objects to XML documents are known in the art and described in the publication"Using XML as an Object Interchange Format" by G. M. Bierman (May 17, 2000), available on the Internet at "http://www.odmg.org/library/readingroom/oifml.pdf", which publication is incorporated herein by reference in its entirety.

For instance, a class PERSON may be defined with attributes NAME and AGE as follows:

```
interface Person {
        attribute string Name;
        attribute unsigned short Age;
};
```

An object instance of the class PERSON may have the name "Sally" and an age of eleven. Below is an example of how the instance of the object "Sally" of the class person may map to a tagged XML file including tagged attribute elements and tagged attribute values.

```
<object oid="Sally">
        <class>Person</class>
        <contents>
                <attribute name="Name">
                        <value><string val="Sally"/></value>
                </attribute>
                <attribute name="Age">
                        <value><unsignedshort val="11"/></value>
                </attribute>
        </contents>
</object>
```

Moreover, when generating the XML document from an object oriented object, the XML translator 16 and 26 would further generate a document type definition (DTD) into the XML document defining the hierarchical organization of attributes, values, and

other elements in the XML document. The XML translators 16 and 26 may receive a schema indicating the class structure of the data object, including all attributes and attribute values for the class. The XML translators 16 and 26 would then be able to generate an XML shell file based on the layout of attributes and values in the class schema. The XML shell file could include the root tag as well as all the class tags, attribute tags, and value tags with no provided values. The XML translators 16 and 26 would then process the objects 14 and 22, respectively, to populate the tagged value fields with the attribute values provided in the objects 14 and 22.

In one implementation, the object oriented XML database 10 is formed in an XML file 10 as shown in FIG. 2, including a root tag 52 and a plurality of XML objects 54a, ... n. Each object is the member of an EMPLOYEE class, having as attributes a NAME of the employee as well as a POSITION attribute. Object 54a is an instance of the class EMPLOYEE, having a NAME value of "Joe Smith" and a "POSITION" value of "Software Engineer". Other objects would include instances of other employees, including the employee name and job position. Each XML object includes the content of an XML document that was generated by the XML translators 16, 26 representing an object in a structured or object oriented language. The OOD APIs 30 are capable of parsing the object oriented XML database (XMLOOD) 10 file to access and perform operations with respect to the XML objects 54a,..., n.

FIG. 3 illustrates logic implemented in the XML translators 16 and 26 to transform an object in a structured or object oriented language into an XML document 18, 28. Control begins at block 100 with the XML translator 16, 26 receiving a call to transform an object to an XML document. In response, the XML translators 16, 26 generate (at block 102) a shell XML document with attribute tags and value tags initialized to empty values based on a schema of the class of which the object is an instance. The schema provides a representation of the attributes of a class and their values. The XML translators 16, 26 further generate (at block 104) a document type definition (DTD) that defines how the markup tags should be interpreted by the

application presenting the document. The XML translators 16, 26 then extract (at block 106) the values for the attributes from the received object. For each attribute value provided in the object, the XML translator 16, 26 inserts (at block 108) the value into the corresponding tagged value field in the XML document 18, 28. The generated XML document 18, 28 is then validated (at block 110) against the generated document type definition (DTD) in a manner known in the art. If (at block 112) the XML file does not validate, then an error is returned (at block 114). Otherwise, the valid XML file is returned (at block 116).

The logic of FIG. 3 may be invoked by an API called by the application 12 to add an object to the object oriented XML database (XMLOOD) 10. The clients 2 and 4 would be provided with client OOD APIs that the applications 12, 20 may call to perform operations with respect to the object oriented XML database (XMLOOD) 10. Such client OOD APIs would generate a stream of data and commands to the database daemon 24 that instruct the database daemon 24 to perform a requested operation against the XMLOOD 10.

FIG. 4 illustrates logic executed in the XML translators 16, 26 to transform an XML document 18, 28 representing an object from the object oriented XML database (XMLOOD) 10 to an object in the structured (e.g., C, FORTRAN, Pascal, etc.) or object oriented programming (e.g., Java, C++, Smalltalk, etc.) language of the applications 12, 20. The logic of FIG. 4 would be invoked by a client OOD API that receives (at block 150) an XML document from the database daemon 24 and calls the XML translator 16, 26 to transform the XML document to an object in the structured or object oriented language of the application 12, 20 to return to the application to use. The XML translator 16, 26 validates (at block 152) the received XML document with the document type definition (DTD) included in the XML document. If (at block 154) the XML document does not validate, then an error is returned (at block 156). Otherwise, if the XML document is valid, then the XML translator 16, 26 generates (at block 158) an instance of the data object for the class based on the class schema initialized with empty

attribute values. The XML translator 16, 26 parses the XML document and extracts (at block 160) attribute values and inserts (at block 162) the extracted values into the generated data object. The data object is returned (at block 164) to the application 12, 20.

FIG. 5 illustrates logic implemented in the database daemon 24 and an ADDOBJECT API that is part of the OOD APIs 30 to add the content of an XML document 18, 28 representing an object 14, 22 to the object oriented XML database 10. At block 200, the database daemon 24 receives an object to add to the XMLOOD 10. If (at block 202) the received object is not an XML document, then an error is returned (at block 204). Otherwise, the database daemon 24 calls the ADDOBJECT API. In response to the call of the ADDOBJECT API (at block 210), an object identifier (OID) is assigned (at block 212) to uniquely identify the object in the XMLOOD 10. The content of the XML document, including the attribute and attribute value information, is then added (at block 214) to the object oriented XML database 10.

FIG. 6 illustrates logic implemented in the database daemon 24 and a GETLIST API that is part of the OOD APIs 30 to generate a list of object identifiers (OIDs) in response (at block 250) to a GETLIST request from one of the applications 12, 20. In response to the GETLIST request, the database daemon 24 calls (at block 252) GETLIST OOD API. The GETLIST API parses (at block 262) the XMLOOD 10 to determine the OIDs from all the XML objects 54a, ..., n and assemble (at block 264) a list of all the OIDs. The GETLIST OOD API would return (at block 256) the list to the database daemon 24, which in turn would return the list to the client 2, 4 applications 12, 20.

FIG. 7 illustrates logic implemented in the database daemon 24 and a GETOBJECT API that is part of the OOD API 30 set to retrieve an object having the specified OID. At block 300, the database daemon 24 receives a GETOBJECT request from one of the applications 12, 20, which is a member of the client OOD APIs. In response, the database daemon 24 calls (at block 302) a GETOBJECT API from the OOD APIs 30 to retrieve the object having the requested OID.

Blocks 310 to 322 illustrate logic implemented in the GETOBJECT API. At block 310, the GETOBJECT API is called. In response, a variable *i* is set (at block 312) to one. The object oriented XML database (XMLOOD) 10 is scanned for the root tag of the *i*th XML object. If (at block 316) the requested OID matches the OID of the *i*th XML object, then a copy of the *i*th XML object is returned (at block 318) to the database daemon 24. Otherwise, if the OIDs do not match and if (at block 320) the *i*th XML object is the last object, then an error is returned (at block 322) indicating that no XML object in the database 10 has the requested OID. If the *i*th object is not the last object, then *i* is incremented (at block 324) and control returns to block 314 to determine if the next object has a matching OID. In response (at block 350) to receiving a matching XML object 54a, ..., n, the database daemon 24 returns the XML object to the requesting application 10, 20.

In the above descriptions, the objects and database of objects are implemented in an XML document. The client applications are provided with an XML translator to translate objects in a structured or object oriented programming language to an XML document and translate XML documents into objects. This allows the representation of objects generated for different structured and/or object oriented programming languages to be stored and manipulated in a same object database. In this way, different applications in different programming languages may share the same object oriented database and data objects therein.

What follows are some alternative implementations.

The preferred embodiments may be implemented as a method, apparatus or article of manufacture using standard programming and/or engineering techniques to produce software, firmware, hardware, or any combination thereof. The term "article of manufacture" as used herein refers to code or logic implemented in hardware logic (e.g., an integrated circuit chip, Field Programmable Gate Array (FPGA), Application Specific Integrated Circuit (ASIC), etc.) or a computer readable medium (e.g., magnetic storage medium (e.g., hard disk drives, floppy disks,, tape, etc.), optical storage (CD-ROMs,

optical disks, etc.), volatile and non-volatile memory devices (e.g., EEPROMs, ROMs, PROMs, RAMs, DRAMs, SRAMs, firmware, programmable logic, etc.). Code in the computer readable medium is accessed and executed by a processor. The code in which preferred embodiments are implemented may further be accessible through a transmission media or from a file server over a network. In such cases, the article of manufacture in which the code is implemented may comprise a transmission media, such as a network transmission line, wireless transmission media, signals propagating through space, radio waves, infrared signals, etc. Of course, those skilled in the art will recognize that many modifications may be made to this configuration without departing from the scope of the present invention, and that the article of manufacture may comprise any information bearing medium known in the art.

In the discussed implementations, one client application comprised a Java application and the other a non-Java application. However, the different client applications may both be implemented in different non-Java object oriented programming languages, or implemented in structured programming languages.

In discussed implementations, the object oriented XML database included only objects that were instances of the same class. In further implementations, objects from multiple classes may be maintained in the object oriented XML database.

In discussed implementations, the XML translators 16, 26 were maintained in the client systems 2, 4. In alternative implementations, the clients 2, 4 may not maintain the XML translator 16, 26. Instead, the database daemon 24 may maintain XML translators to translate objects in different structured or object oriented programming languages into an XML document, and to translate XML objects to objects in the structured or object oriented programming language of the requesting client application.

In discussed implementations, an XML document format was used to transport and store the objects in the persistent object oriented database. In alternative implementations, a different file format, such as a different structured file format, may be used to represent the attributes and values of the object instance of the class, such as a

different standard generalized markup language (SGML), hypertext markup language (HTML), extensible hypertext markup language (xHTML), etc. In this way, the attribute of an object may be represented in alternative structured document formats.

In discussed implementations, the object oriented database comprises an XML document. In alternative implementations, the content of the XML objects, or other structured document objects, may be stored in data structures other than an XML document, such as a database of objects or files, an index of files, an object oriented database, etc. Further, the structured document formats used to represent the objects and implement the database may comprise different file formats.

In discussed implementations, the client applications are included in client systems that communicate with the database server over a network. In alternative implementations, the applications and XML translators may be implemented on the same computing platform including the database daemon.

The foregoing description of various implementation of the invention has been presented for the purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise form disclosed. Many modifications and variations are possible in light of the above teaching. It is intended that the scope of the invention be limited not by this detailed description, but rather by the claims appended hereto. The above specification, examples and data provide a complete description of the manufacture and use of the composition of the invention. Since many embodiments of the invention can be made without departing from the spirit and scope of the invention, the invention resides in the claims hereinafter appended.

** JAVA is a trademark of Sun Microsystems, Inc.

WHAT IS CLAIMED IS:

1. A method for maintaining a database of objects, comprising:

receiving at least one structured document representing an instance of an object including attributes and attribute values defined for a class; and

adding content of the structured document representing the object into the database, wherein the database is capable of storing multiple structured documents representing multiple objects.

2. The method of claim 1, further comprising:

receiving multiple structured documents representing instances of objects defined for the class, wherein the objects represented in at least two different received structured documents were generated in different programming languages.

3. The method of claim 2, wherein application programs implemented in different programming languages can share objects represented as structured documents in the database.

4. The method of claim 1, wherein the database comprises a structured document, and wherein adding the content of each structured document representing one object comprises inserting the content of the structured document representing the object into the structured document implementing the database.

5. The method of claim 4, wherein the database structured document and the structured documents representing the objects are in a same file format.

6. The method of claim 5, wherein the same file format comprises an extensible markup language (XML) document format.

7. The method of claim 1, wherein the structured document comprises an extensible markup language (XML) document.

8. The method of claim 1, wherein all the objects represented as structured document content in the database are instantiated from a same class.

9. A method for accessing a database of objects, comprising:

generating an instance of at least one object including attributes and attribute values defined for a class;

for each generated object, generating a structured document representing the object and including a representation of the attributes and attribute values in the object; and

transferring each structured document to the database to maintain.

10. The method of claim 9, further comprising:

receiving a structured document from the database representing attributes and attribute values for one object; and

generating an object including the attributes and attribute values represented in the structured document, wherein the generated object embodies the object represented by the received structured document.

11. The method of claim 9, wherein generating the at least one object further comprises:

(i) generating an instance of a first object including attributes and attribute values defined for the class in a first programming language; and

(ii) generating an instance of a second object including attributes and attribute values defined for the class in a second programming language;

wherein generating one structured document for each generated object further comprises:

(i) generating a first structured document representing the first object; and

(ii) generating a second structured document representing the second object.

12. The method of claim 11, wherein application programs implemented in the first and second programming languages are capable of sharing objects represented as structured documents in the database.

13. The method of claim 9, wherein the database comprises a structured document, and wherein adding the content of the structured documents representing the objects comprises inserting the content into the database structured document.

14. The method of claim 13, wherein the database structured document and the structured documents representing the objects are in a same file format.

15. The method of claim 14, wherein the same file format comprises an extensible markup language (XML) document format.

16. The method of claim 9, wherein the structured document comprises an extensible markup language (XML) document.

17. The method of claim 9, wherein all the objects represented as structured document content in the database are instantiated from a same class.

18. A system for maintaining a database of objects, comprising:

means for receiving at least one structured document representing an instance of an object including attributes and attribute values defined for a class; and

means for adding content of the structured document representing the object into the database, wherein the database is capable of storing multiple structured documents representing multiple objects.

19. The system of claim 18, further comprising:

means for receiving multiple structured documents representing instances of objects defined for the class, wherein the objects represented in at least two different received structured documents were generated in different programming languages.

20. The system of claim 19, wherein application programs implemented in different programming languages can share objects represented as structured documents in the database.

21. The system of claim 18, wherein the database comprises a structured document, and wherein adding the content of each structured document representing one object comprises inserting the content of the structured document representing the object into the structured document implementing the database.

22. The system of claim 21, wherein the database structured document and the structured documents representing the objects are in a same file format.

23. The system of claim 22, wherein the same file format comprises an extensible markup language (XML) document format.

24. The system of claim 18, wherein the structured document comprises an extensible markup language (XML) document.

25. The system of claim 18, wherein all the objects represented as structured document content in the database are instantiated from a same class.

26. A system for accessing a database of objects, comprising:

means for generating an instance of at least one object including attributes and attribute values defined for a class;

means for generating, for each generated object, a structured document representing the object and including a representation of the attributes and attribute values in the object; and

means for transferring each structured document to the database to maintain.

27. The system of claim 26, further comprising:

means for receiving a structured document from the database representing attributes and attribute values for one object; and

means for generating an object including the attributes and attribute values represented in the structured document, wherein the generated object embodies the object represented by the received structured document.

28. The system of claim 26, wherein the means for generating the at least one object further performs:

(i) generating an instance of a first object including attributes and attribute values defined for the class in a first programming language; and

(ii) generating an instance of a second object including attributes and attribute values defined for the class in a second programming language;

wherein the means for generating one structured document for each generated object further performs:

(i) generating a first structured document representing the first object; and

(ii) generating a second structured document representing the second object.

29. The system of claim 28, wherein application programs implemented in the first and second programming languages are capable of sharing objects represented as structured documents in the database.

30. The system of claim 26, wherein the database comprises a structured document, and wherein adding the content of the structured documents representing the objects comprises inserting the content into the database structured document.

31. The system of claim 30, wherein the database structured document and the structured documents representing the objects are in a same file format.

32. The system of claim 31, wherein the same file format comprises an extensible markup language (XML) document format.

33. The system of claim 26, wherein the structured document comprises an extensible markup language (XML) document.

34. The system of claim 26, wherein all the objects represented as structured document content in the database are instantiated from a same class.

35. An article of manufacture for maintaining a database of objects, wherein the article of manufacture comprises code implemented in a computer readable medium capable of causing a processor to perform:

receiving at least one structured document representing an instance of an object including attributes and attribute values defined for a class; and

adding content of the structured document representing the object into the database, wherein the database is capable of storing multiple structured documents representing multiple objects.

36. The article of manufacture of claim 35, wherein the code is further capable of causing the processor to perform:

receiving multiple structured documents representing instances of objects defined for the class, wherein the objects represented in at least two different received structured documents were generated in different programming languages.

37. The article of manufacture of claim 36, wherein application programs implemented in different programming languages can share objects represented as structured documents in the database.

38. The article of manufacture of claim 35, wherein the database comprises a structured document, and wherein adding the content of each structured document representing one object comprises inserting the content of the structured document representing the object into the structured document implementing the database.

39. The article of manufacture of claim 38, wherein the database structured document and the structured documents representing the objects are in a same file format.

40. The article of manufacture of claim 39, wherein the same file format comprises an extensible markup language (XML) document format.

41. The article of manufacture of claim 35, wherein the structured document comprises an extensible markup language (XML) document.

42. The article of manufacture of claim 35, wherein all the objects represented as structured document content in the database are instantiated from a same class.

43. An article of manufacture for accessing a database of objects, wherein the article of manufacture comprises code implemented in a computer readable medium capable of causing a processor to perform:

generating an instance of at least one object including attributes and attribute values defined for a class;

for each generated object, generating a structured document representing the object and including a representation of the attributes and attribute values in the object; and

transferring each structured document to the database to maintain.

44. The article of manufacture of claim 43, wherein the code is further capable of causing the processor to perform:

receiving a structured document from the database representing attributes and attribute values for one object; and

generating an object including the attributes and attribute values represented in the structured document, wherein the generated object embodies the object represented by the received structured document.

45. The article of manufacture of claim 43, wherein generating the at least one object further comprises:

(i) generating an instance of a first object including attributes and attribute values defined for the class in a first programming language; and

(ii) generating an instance of a second object including attributes and attribute values defined for the class in a second programming language;

wherein generating one structured document for each generated object further comprises:

(i) generating a first structured document representing the first object; and

(ii) generating a second structured document representing the second object.

46. The article of manufacture of claim 45, wherein application programs implemented in the first and second programming languages are capable of sharing objects represented as structured documents in the database.

47. The article of manufacture of claim 43, wherein the database comprises a structured document, and wherein adding the content of the structured documents representing the objects comprises inserting the content into the database structured document.

48. The article of manufacture of claim 47, wherein the database structured document and the structured documents representing the objects are in a same file format.

49. The article of manufacture of claim 48, wherein the same file format comprises an extensible markup language (XML) document format.

50. The article of manufacture of claim 43, wherein the structured document comprises an extensible markup language (XML) document.

51. The article of manufacture of claim 43, wherein all the objects represented as structured document content in the database are instantiated from a same class.

52. A computer readable medium including a computer database of objects, comprising:

at least one structured document representing an instance of an object including attributes and attribute values defined for a class, wherein the database is capable of storing multiple structured documents representing multiple objects.

53. The computer readable medium of claim 52, wherein the database stores multiple structured documents representing instances of objects defined for the class, and wherein the objects represented in at least two different structured documents stored in the database were generated in different programming languages.

54. The computer readable medium of claim 53, wherein application programs implemented in different programming languages can share objects represented as structured documents in the database.

55. The computer readable medium of claim 52, wherein the database comprises a structured document, and wherein structured documents representing objects are added to the database by inserting the content of the structured document representing the object into the structured document implementing the database.

56. The computer readable medium of claim 55, wherein the database structured document and the structured documents representing the objects are in a same file format.

57. The computer readable medium of claim 56, wherein the same file format comprises an extensible markup language (XML) document format.

58. The computer readable medium of claim 52, wherein the structured document comprises an extensible markup language (XML) document.

59. The computer readable medium of claim 52, wherein all the objects represented as structured document content in the database are instantiated from a same class.

Express Mail No. EL414495378US
Docket No. P5635
Firm No. 0045.0005

METHOD, SYSTEM, PROGRAM, AND COMPUTER READABLE MEDIUM FOR PROVIDING A DATABASE FOR OBJECT ORIENTED OBJECTS

ABSTRACT

Provided is a method, system, program, and computer readable medium for maintaining a database of objects. At least one structured document is received representing an instance of an object including attributes and attribute values defined for a class. Content of the structured document representing the object is added into the database, wherein the database is capable of storing multiple structured documents representing multiple objects. In another implementation, an instance of at least one object including attributes and attribute values defined for a class is generated. For each generated object, a structured document is generated representing the object and including a representation of the attributes and attribute values in the object. Each structured document is transferred to the database to maintain.

FIG. 1

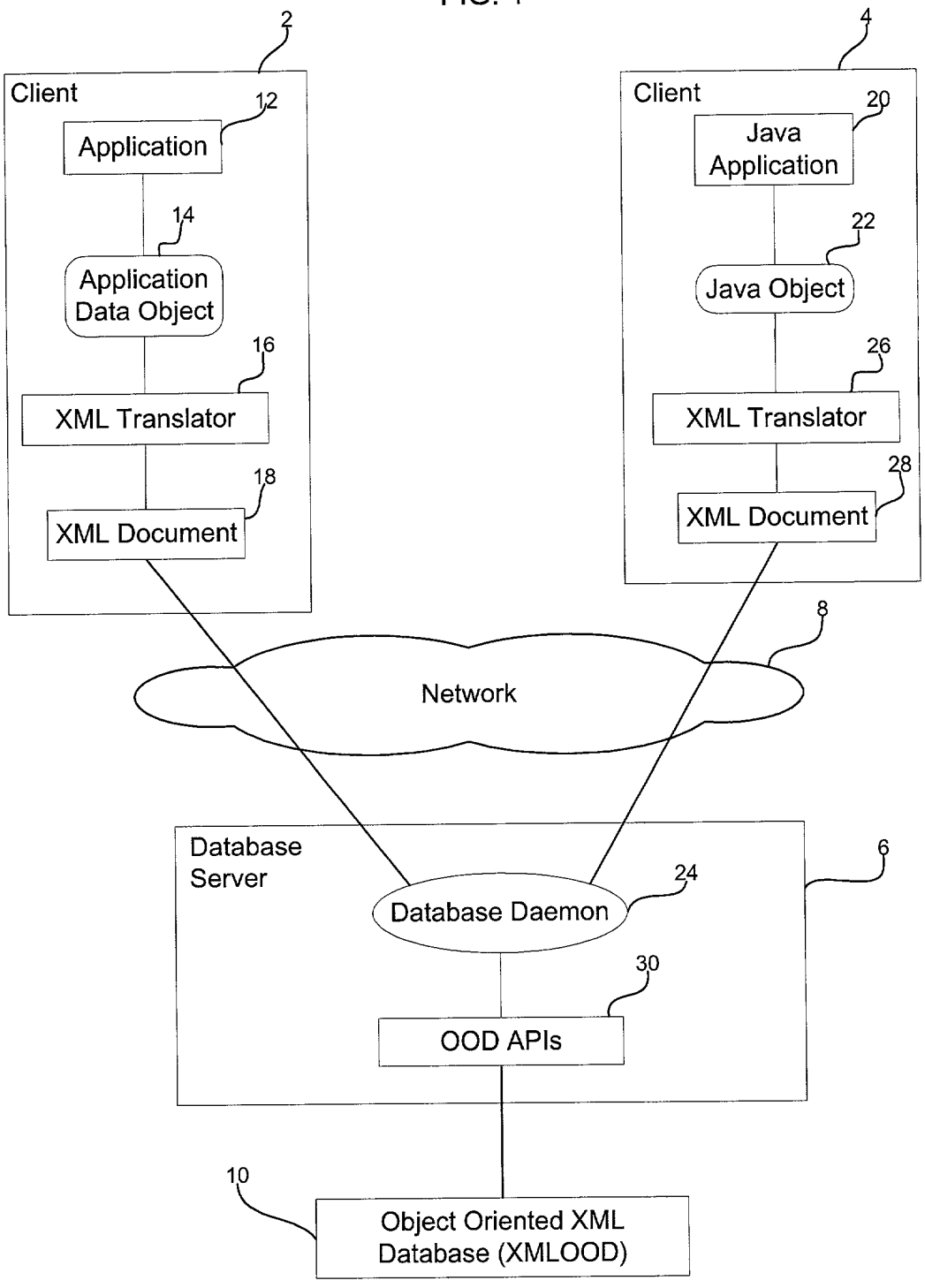
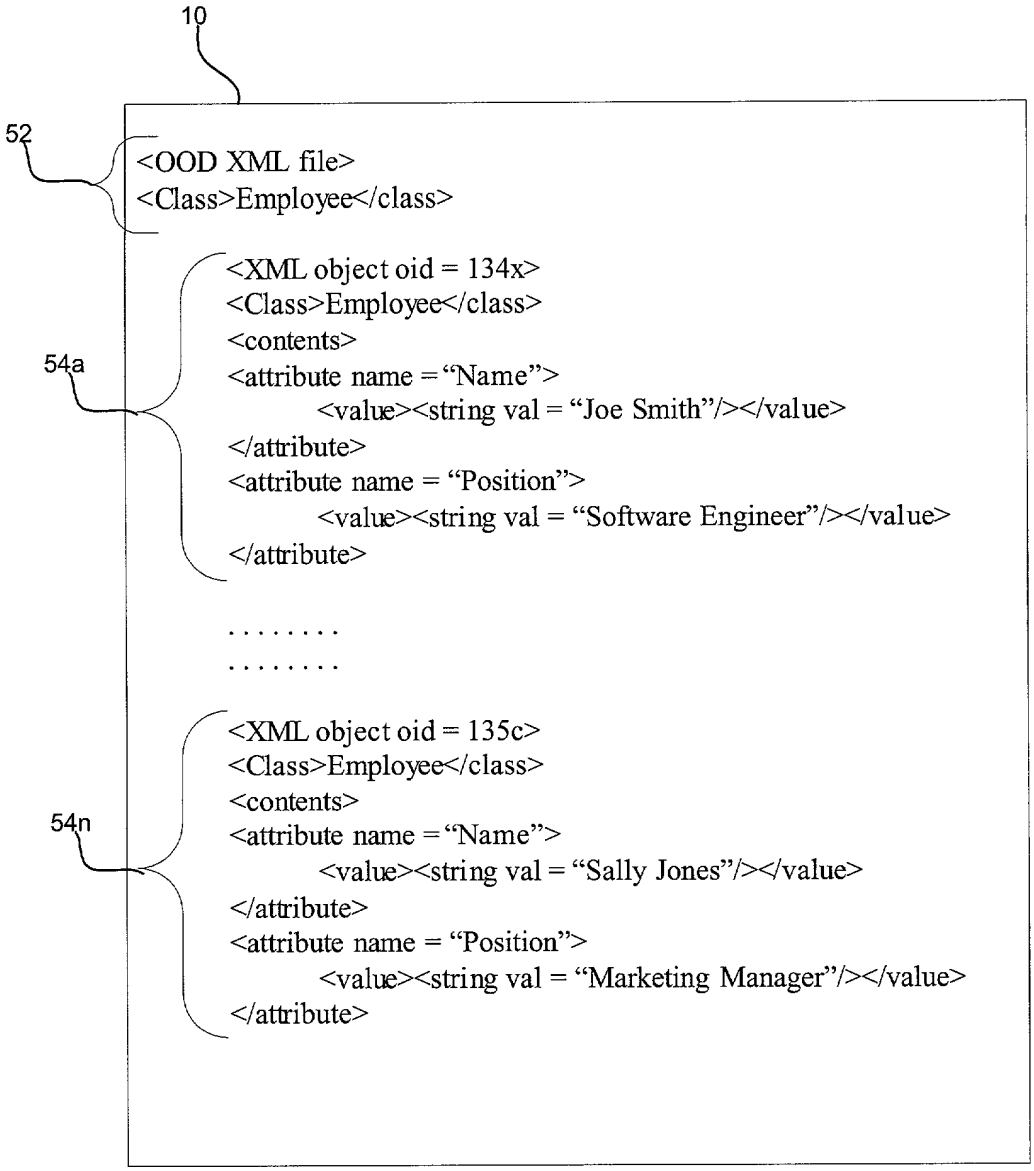

2
Client
12
Application
14
Application Data Object
16
XML Translator
18
XML Document
4
Client
20
Java Application
22
Java Object
26
XML Translator
28
XML Document
8
Network
Database Server
6
24
Database Daemon
30
OOD APIs
10
Object Oriented XML Database (XMLOOD)

FIG. 2


10
52
<OOD XML file>
<Class>Employee</class>
54a
<XML object oid = 134x>
<Class>Employee</class>
<contents>
<attribute name = "Name">
<value><string val = "Joe Smith"/></value>
</attribute>
<attribute name = "Position">
<value><string val = "Software Engineer"/></value>
</attribute>
. . . . . .
. . . . . .
54n
<XML object oid = 135c>
<Class>Employee</class>
<contents>
<attribute name = "Name">
<value><string val = "Sally Jones"/></value>
</attribute>
<attribute name = "Position">
<value><string val = "Marketing Manager"/></value>
</attribute>

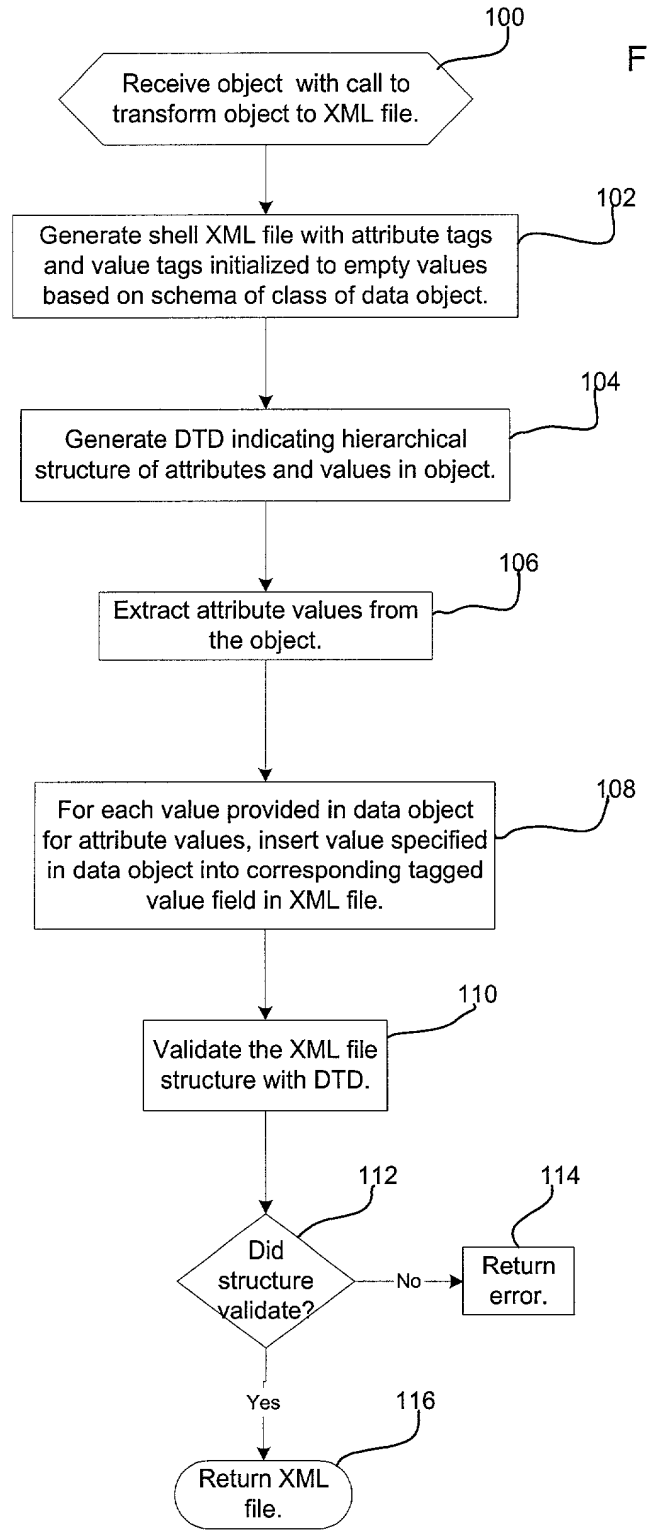

100
Receive object with call to transform object to XML file.
102
Generate shell XML file with attribute tags and value tags initialized to empty values based on schema of class of data object.
104
Generate DTD indicating hierarchical structure of attributes and values in object.
106
Extract attribute values from the object.
108
For each value provided in data object for attribute values, insert value specified in data object into corresponding tagged value field in XML file.
110
Validate the XML file structure with DTD.
112
Did structure validate?
No
114
Return error.
Yes
116
Return XML file.

FIG. 3

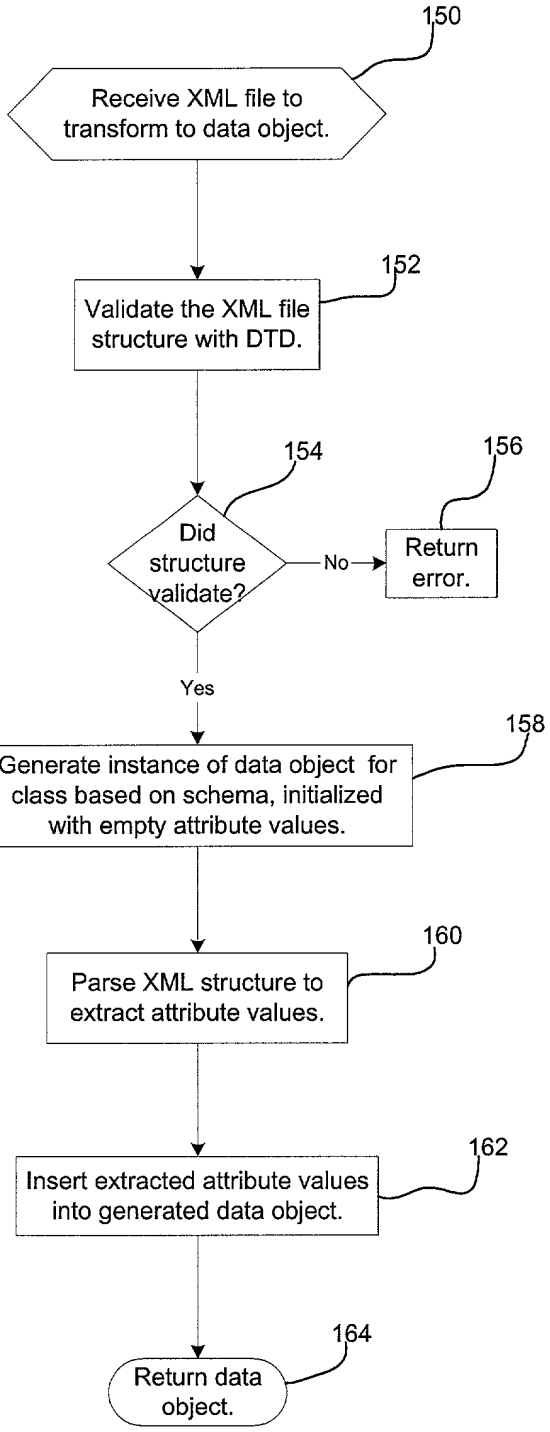

150
Receive XML file to transform to data object.
152
Validate the XML file structure with DTD.
154
Did structure validate?
No
156
Return error.
Yes
158
Generate instance of data object for class based on schema, initialized with empty attribute values.
160
Parse XML structure to extract attribute values.
162
Insert extracted attribute values into generated data object.
164
Return data object.

FIG. 4

FIG. 5

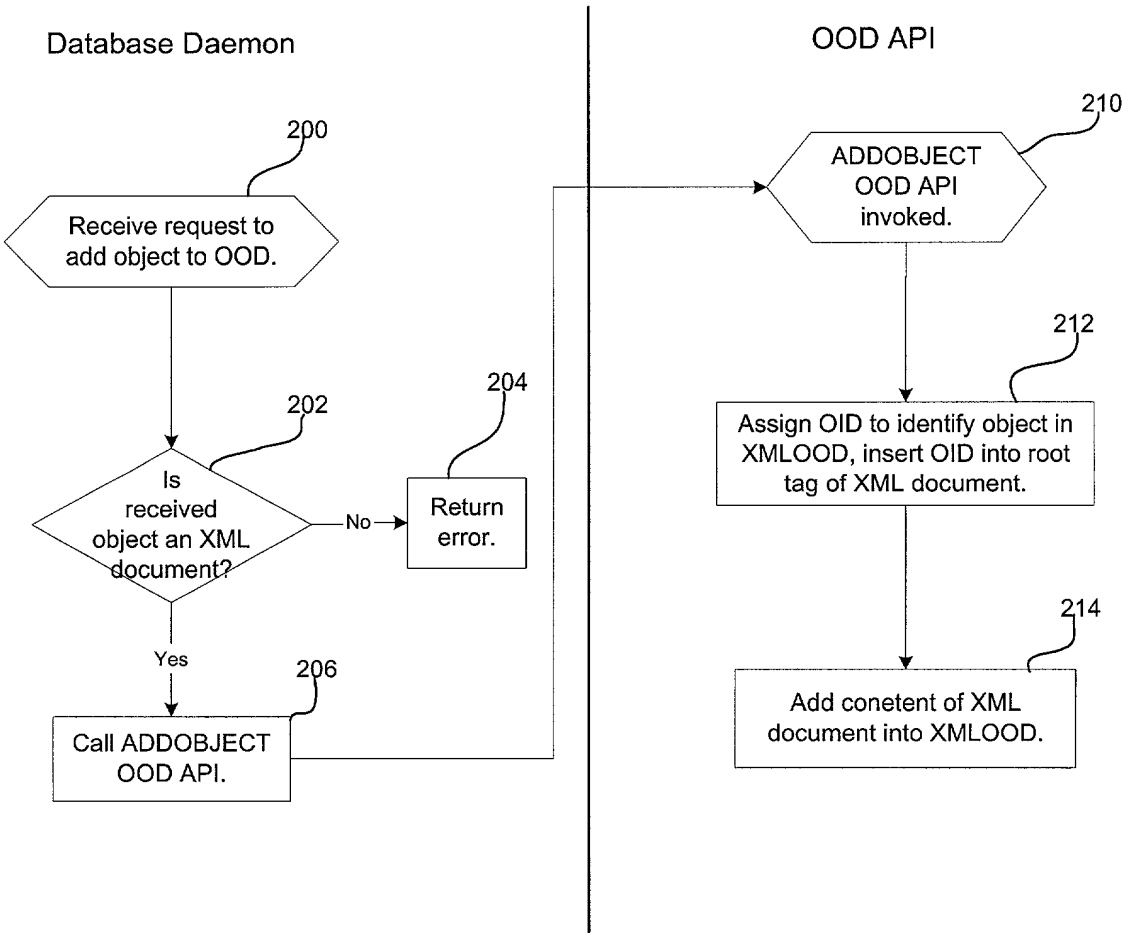

Database Daemon
OOD API
200
Receive request to add object to OOD.
202
Is received object an XML document?
No
204
Return error.
Yes
206
Call ADDOBJECT OOD API.
210
ADDOBJECT OOD API invoked.
212
Assign OID to identify object in XMLOOD, insert OID into root tag of XML document.
214
Add conetent of XML document into XMLOOD.

FIG. 6

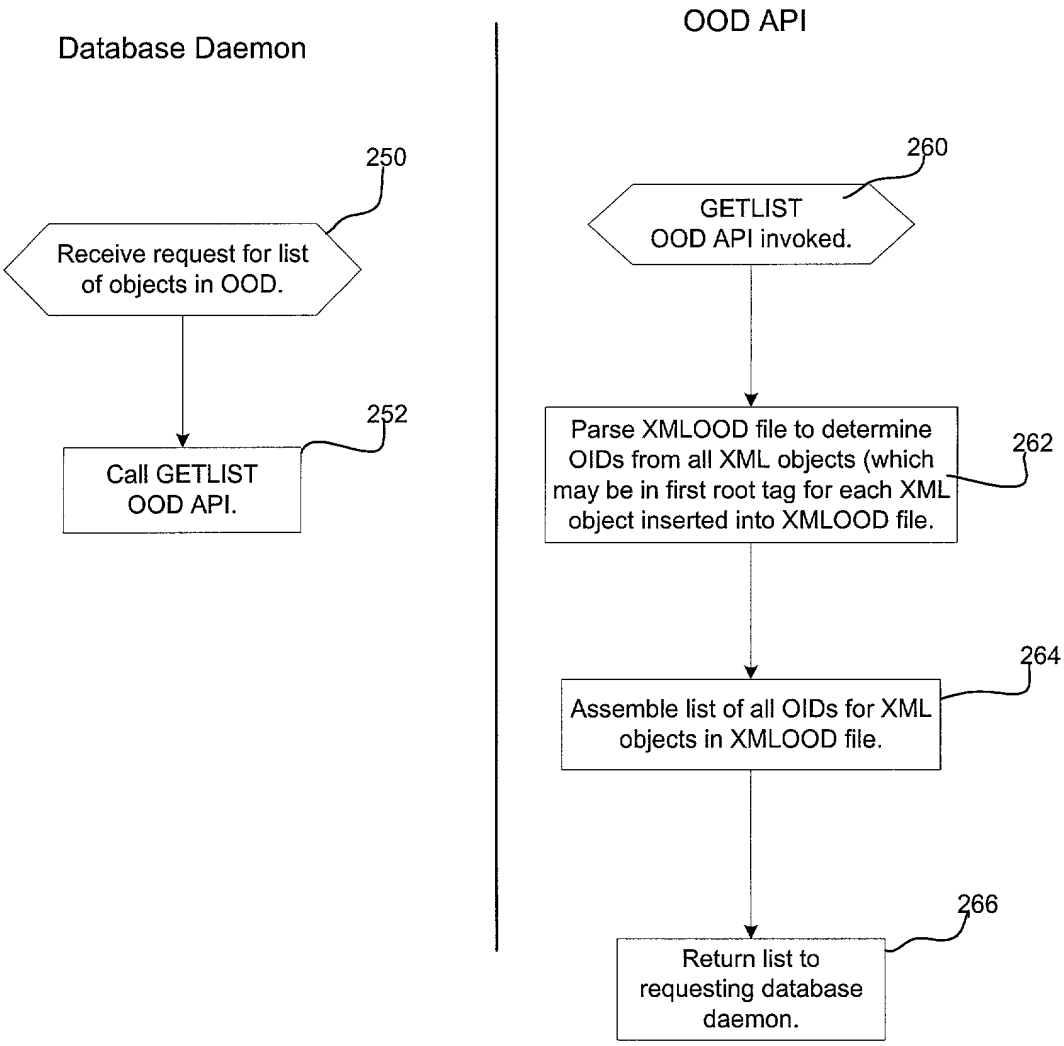
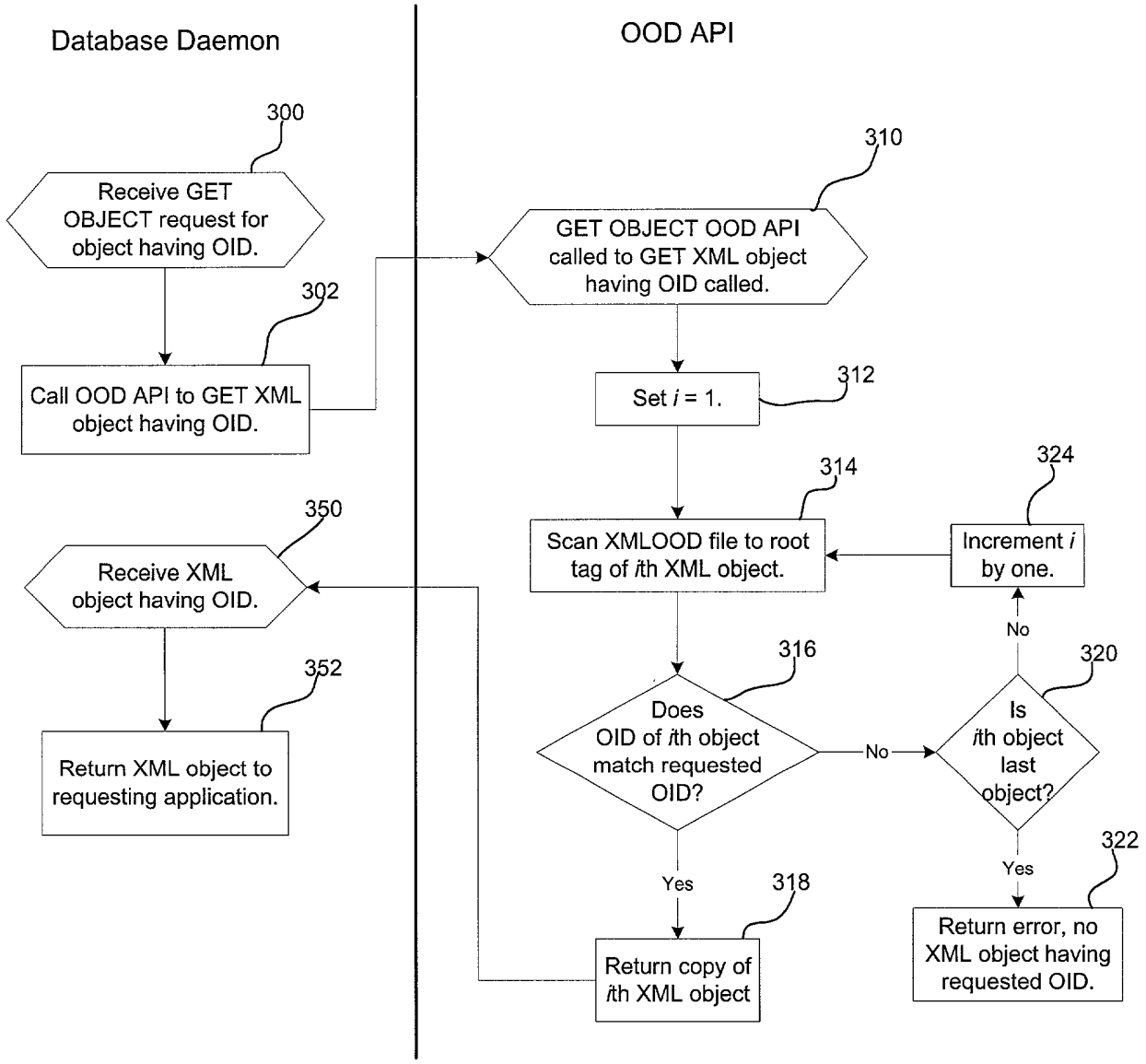

Database Daemon
OOD API
250
Receive request for list of objects in OOD.
252
Call GETLIST OOD API.
260
GETLIST OOD API invoked.
Parse XMLOOD file to determine OIDs from all XML objects (which may be in first root tag for each XML object inserted into XMLOOD file.
262
Assemble list of all OIDs for XML objects in XMLOOD file.
264
Return list to requesting database daemon.
266

FIG. 7


Database Daemon
OOD API
300
Receive GET OBJECT request for object having OID.
302
Call OOD API to GET XML object having OID.
310
GET OBJECT OOD API called to GET XML object having OID called.
312
Set i = 1.
314
Scan XMLOOD file to root tag of ith XML object.
316
Does OID of ith object match requested OID?
No
320
Is ith object last object?
No
324
Increment i by one.
Yes
318
Return copy of ith XML object
Yes
322
Return error, no XML object having requested OID.
350
Receive XML object having OID.
352
Return XML object to requesting application.

DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION **DOCKET: P5635**

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name;

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled

METHOD, SYSTEM, PROGRAM, AND COMPUTER READABLE MEDIUM FOR PROVIDING A DATABASE FOR OBJECT ORIENTED OBJECTS

the specification of which (check one)

X is attached hereto.
___ was filed on _____
as Application Serial No. _____
and was amended on _____ (if applicable).

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to patentability as defined in Title 37, Code of Federal Regulations, Section 1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, Section 119(a)-(d) or Section 365(b) of any foreign application(s) for patent or inventor's certificate, or Section 365(a) of any PCT International application which designated at least one country other than the United States, listed below and have also identified below any foreign application for patent or inventor's certificate or PCT International application having a filing date before that of the application on which priority is claimed:

Prior Foreign Application(s)			Priority Claimed
None			_Yes _No
(Number)	(Country)	(Day/Month/Year Filed)	

I hereby claim the benefit under Title 35, United States Code, Section 120 of any United States application(s) or Section 365(c) of any PCT International application designating the United States, listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States or PCT International application in the manner provided by the first paragraph of Title 35, United States Code, Section 112, I acknowledge the duty to disclose information which is material to patentability as defined in Title 37, Code of Federal Regulations, Section 1.56, which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

None		
(Application Serial No.)	(Filing Date)	(Status) (patented, pending, abandoned)

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION **DOCKET: P5635**

POWER OF ATTORNEY: David W. Victor, Reg. No. 39,867; William K. Konrad, Reg. No. 28,868; Gary D. Mann, Reg. No. 34,867; Alan S. Raynes, Reg. No. 39,809; Richard K. Yoon, Reg. No. 42,247; Kenneth Olsen, Reg. No. 26,493; Timothy J. Crean, Reg. No. 37,116; Robert S. Hauser, Reg. No. 37,847; Joseph T. FitzGerald, Reg. No. 33,881; Alexander E. Silverman, Reg. No. 37,940; Anirma R. Gupta, Reg. No.38,275; Sean P. Lewis, Reg. No. 42,798; Michael Schallop, Reg. No. 44,319; Bernice B. Chen, Reg. No. 42,403; Kenta Suzue, Reg. No. 45,145; Noreen Krall, Reg. No.39,734; Richard J. Lutton, Reg. No. 39,756; Monica L. Ward, Reg. No.40,696; Marc D. Foodman, Reg. No. 34,110; Naren Chaganti, Reg. No. 44,602; Elaine Lee, Reg. No. 41,936; Hugh Matsubayashi, Reg. No. 43,779.

Send correspondence to:

David Victor, Esq
1180 South Beverly Dr., Ste. 501
Los Angeles, CA 90035

Direct all telephone calls to David Victor at (310) 553-7977

FULL NAME OF INVENTOR ONE: Terence Leong	
INVENTORS SIGNATURE: Terence Leong	DATE: 11/22/00
RESIDENCE: 812 Oxford Way, Benicia, California 94510	
CITIZENSHIP: United States	
POST OFFICE ADDRESS: same as residence	

FULL NAME OF INVENTOR TWO: Mahima Mallikarjuna	
INVENTORS SIGNATURE:	DATE: 11/22/2000
RESIDENCE: 38660 Lexington Street, #521, Fremont, California 94536	
CITIZENSHIP: India	
POST OFFICE ADDRESS: same as residence	

FULL NAME OF INVENTOR THREE: Julian Taylor	
INVENTORS SIGNATURE:	DATE:
RESIDENCE: P.O. Box 3198, 33 Ute Way, Nederland, Colorado 80466	
CITIZENSHIP: United States	
POST OFFICE ADDRESS: same as residence	

DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION **DOCKET: P5635**

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name;

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled

METHOD, SYSTEM, PROGRAM, AND COMPUTER READABLE MEDIUM FOR PROVIDING A DATABASE FOR OBJECT ORIENTED OBJECTS

the specification of which (check one)

X is attached hereto.
___ was filed on _____
as Application Serial No. _____
and was amended on _____ (if applicable).

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to patentability as defined in Title 37, Code of Federal Regulations, Section 1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, Section 119(a)-(d) or Section 365(b) of any foreign application(s) for patent or inventor's certificate, or Section 365(a) of any PCT International application which designated at least one country other than the United States, listed below and have also identified below any foreign application for patent or inventor's certificate or PCT International application having a filing date before that of the application on which priority is claimed:

Prior Foreign Application(s)			Priority Claimed
None			_Yes _No
(Number)	(Country)	(Day/Month/Year Filed)	

I hereby claim the benefit under Title 35, United States Code, Section 120 of any United States application(s) or Section 365(c) of any PCT International application designating the United States, listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States or PCT International application in the manner provided by the first paragraph of Title 35, United States Code, Section 112, I acknowledge the duty to disclose information which is material to patentability as defined in Title 37, Code of Federal Regulations, Section 1.56, which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

None		
(Application Serial No.)	(Filing Date)	(Status) (patented, pending, abandoned)

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION **DOCKET: P5635**

POWER OF ATTORNEY: David W. Victor, Reg. No. 39,867; William K. Konrad, Reg. No. 28,868; Gary D. Mann, Reg. No. 34,867; Alan S. Raynes, Reg. No. 39,809; Richard K. Yoon, Reg. No. 42,247; Kenneth Olsen, Reg. No. 26,493; Timothy J. Crean, Reg. No. 37,116; Robert S. Hauser, Reg. No. 37,847; Joseph T. FitzGerald, Reg. No. 33,881; Alexander E. Silverman, Reg. No. 37,940; Anirma R. Gupta, Reg. No.38,275; Sean P. Lewis, Reg. No. 42,798; Michael Schallop, Reg. No. 44,319; Bernice B. Chen, Reg. No. 42,403; Kenta Suzue, Reg. No. 45,145; Noreen Krall, Reg. No.39,734; Richard J. Lutton, Reg. No. 39,756; Monica L. Ward, Reg. No.40,696; Marc D. Foodman, Reg. No. 34,110; Naren Chaganti, Reg. No. 44,602; Elaine Lee, Reg. No. 41,936; Hugh Matsubayashi, Reg. No. 43,779.

Send correspondence to:

David Victor, Esq
1180 South Beverly Dr., Ste. 501
Los Angeles, CA 90035

Direct all telephone calls to David Victor at (310) 553-7977

FULL NAME OF INVENTOR ONE: Terence Leong
INVENTORS SIGNATURE: DATE:
RESIDENCE: 812 Oxford Way, Benicia, California 94510
CITIZENSHIP: United States
POST OFFICE ADDRESS: same as residence

FULL NAME OF INVENTOR TWO: Mahima Mallikarjuna
INVENTORS SIGNATURE: DATE:
RESIDENCE: 38660 Lexington Street, #521, Fremont, California 94536
CITIZENSHIP: India
POST OFFICE ADDRESS: same as residence

FULL NAME OF INVENTOR THREE: Julian Taylor
INVENTORS SIGNATURE: DATE: November 27, 2000
RESIDENCE: P.O. Box 3198, 53 Ute Way, Nederland, Colorado 80466
CITIZENSHIP: United States
POST OFFICE ADDRESS: same as residence